

Combination of Keyword Extraction with Text Classification by String Vector based KNN

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we proposed the KNN version which receives a string vector, instead of a numerical vector as the approach to the keyword extraction and the text classification, in order to automate fully the keyword extraction. In the previous work, the KNN version was proposed as an approach to the keyword extraction, but a domain should be presented as a tag in preparing a text as the input. In this research, the KNN algorithm is also modified into the string vector-based version and applied to both tasks. In the proposed system, a text without its domain tag is given as the input, its domain is automatically assigned to it through the text classification, and its keywords are extracted by the classifier which corresponds to the domain. The goal of this research is to automate the keyword extraction without presenting a domain tag and to improve the performance of both tasks, using the string vector based KNN version.

I. INTRODUCTION

The keyword extraction is the process of extracting words which represent an entire text. The task is interpreted into a binary classification where a word is classified into keyword or non-keyword. The process of keyword extraction is to index a text which is given as the input into words, classify each word into keyword or non-keyword, and extract words which are classified into keyword as the output. The classification which is mapped from the keyword extraction belongs to the domain dependent task where a same word which is classified into keyword may be classified into non-keyword in another domain. The text domain should be presented as a tag for nominating the classifier which corresponds to the domain.

This research is motivated by the fact that we should know the domain of an input text in advance for executing the keyword extraction. The results from applying the string vector-based version of the KNN algorithm to the keyword extraction were successful. Because the keyword extraction is mapped into a domain dependent classification, the domain must be presented manually for selecting the corresponding classifier. Another motivation is that the results from applying the proposed version to the text classification were successful. The keyword extraction relates to the text classification in order to assign the domain to the text automatically, before executing the keyword extraction.

The string vector based KNN algorithm is adopted for implementing the keyword extraction system which relates to the text categorization. The semantic similarity on string vectors is defined as the operation on them, and the KNN algorithm is modified into the string vector-based version as the approach to both tasks. The modified KNN algorithm is applied to the text categorization which labels the input text automatically with its down domain. In the process of keyword extraction, an input text is indexed into a list of words, each word is classified into keyword or non-keyword by the modified KNN algorithm, and ones which are classified into keyword are extracted as the output. The keyword extraction relates to the text categorization for presenting only a text without its own domain for the keyword extraction system.

Let us mention some merits of this research. Encoding words and texts into string vectors is the solution to the problems in encoding them into numerical vectors. The performance of the keyword extraction and the text classification is improved by adopting the string vector based KNN algorithm. There is no need to present the domain to the keyword extraction, which is given as a domain dependent task, by connecting it with the text classification. We automate the process of deciding the domain to the input text, as the upgrade of the keyword extraction system.

Let us mention some remaining tasks for upgrading this research. We will define and characterize mathematically more semantic operations on string vectors. Words and texts are encoded into compound representations, each of which consists of more than one structured form. Other machine learning algorithms such as the Naïve Bayes and the SVM (Support Vector Machine) and deep learning algorithms such as Restricted Boltzmann Machine and Recurrent Networks may be modified into string vector-based versions. The keyword extraction system may be implemented as a real program or a library by adopting the proposed KNN algorithm.

II. PROPOSED APPROACH

A. Similarity Metric

This section is concerned with the string vector as the representation of a word and the similarity between string vectors. The string vector is defined as a finite ordered set of strings; in the vector, the numerical values are replaced by the strings. It is required to define a similarity matrix for computing the similarity between elements. The similarity between string vectors is the average over one-to-one similarities between their elements. This section is intended to describe the process of encoding a word into a string vector and the process of computing the similarity between string vectors.

The process of encoding words into string vectors is illustrated in Figure 1. In the word-text association, a text collection is constructed by gathering texts, and each word is associated with its own texts based on their information. The string vector features are defined as symbolic forms manually in the feature definition. In the feature value assignment, the string vector which represents a word is filled with text identifiers which correspond to the features. Refer to [1] for studying the encoding process in more detail.

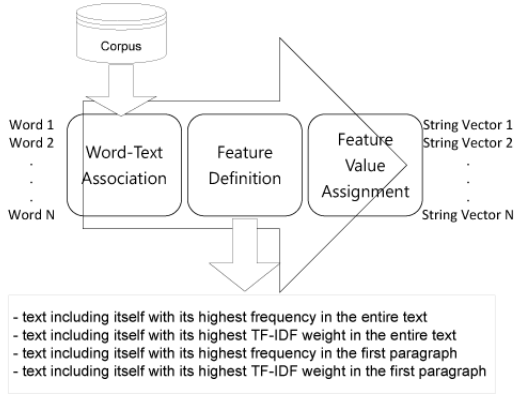


Figure 1. Process of Encoding Words into String Vectors

The similarity matrix which is the basis for computing the similarity between string vectors is illustrated in figure 00. In the matrix, each row and each column correspond to text identifiers, and each element is the similarity between texts, d_i and d_j , as shown in equation (1),

$$s_{ij} = \text{sim}(d_i, d_j) \quad (1)$$

The similarity between two texts, d_i and d_j , is computed by equation (2),

$$\text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|} \quad (2)$$

where D_i is the set of words in the text, d_i , and D_j is the set of words in the text, d_j . The value of similarity

between texts, d_i and d_j , is always given as a normalized value between zero and one, as shown in equation (3)

$$0 \leq \text{sim}(d_i, d_j) \leq 1 \quad (3)$$

The similarity matrix is used for computing the similarity between string vectors which represent words as a reference table.

$$\begin{bmatrix} s_{11} & s_{12} & \dots & s_{1N} \\ s_{21} & s_{22} & \dots & s_{2N} \\ \dots & \dots & \dots & \dots \\ s_{N1} & s_{N2} & \dots & s_{NN} \end{bmatrix} \quad \text{sim}(d_i, d_j) = \frac{2|D_i \cap D_j|}{|D_i| + |D_j|}$$

$0 \leq \text{sim}(d_i, d_j) \leq 1.0$

Figure 2. Semantic Similarity Matrix

Let us mention the process of computing the similarity between string vectors as a semantic similarity between words. The two string vectors which represent words are notated by $\text{str}_1 = [d_{11} \ d_{12} \ \dots \ d_{1d}]$ and $\text{str}_2 = [d_{21} \ d_{22} \ \dots \ d_{2d}]$. The similarity between two string vectors is computed by equation (4),

$$\text{sim}(\text{str}_1, \text{str}_2) = \frac{1}{d} \sum_{i=1}^d \text{sim}(d_{1i}, d_{2i}). \quad (4)$$

The similarity between elements, $\text{sim}(d_{1i}, d_{2i})$, is taken from the similarity which was mentioned above. The value which is generated by equation (4) is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(\text{str}_1, \text{str}_2) \leq 1. \quad (5)$$

Let us make some remarks on the process of encoding words into string vectors and computing the similarity between words. A word is encoded into a string vector by the word-text association, the feature definition, and the feature value assignment. The similarity matrix is defined as the basis for computing the similarity between string vectors. The similarity is computed by equation (4). In the future research, we consider representing a word into multiple string vectors.

B. String Vector based KNN Algorithm

This section is concerned with the string vector based KNN algorithm. It was initially proposed as the approach to the text summarization by Jo in 2018 [1]. The similarity metric between string vectors which was described in the previous section is used for computing the similarity between a novice string vector and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 00. The labeled words which are gathered as samples are encoded into string vectors, and they are notated by a set, $Tr = \{\mathbf{str}_1, \mathbf{str}_2, \dots, \mathbf{str}_N\}$. A novice word is encoded into a string vector notated by $\mathbf{str}$. Its similarities with the string vectors which represent the sample words are computed by equation (4), as $\text{sim}(\mathbf{str}, \mathbf{str}_1), \text{sim}(\mathbf{str}, \mathbf{str}_2), \dots, \text{sim}(\mathbf{str}, \mathbf{str}_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

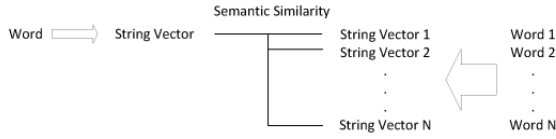


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, \mathbf{str}), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{\mathbf{str}_1^{\text{near}}, \mathbf{str}_2^{\text{near}}, \dots, \mathbf{str}_k^{\text{near}}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

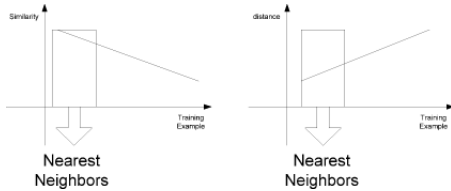


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(\mathbf{str}_i^{\text{near}})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, $\mathbf{str}$, is decided by equation (8),

$$\text{label}(\mathbf{str}) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.



Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the string vector-based version of KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the keyword extraction system which adopts the string vector based KNN algorithm. In Section II-B, we described the proposed version of KNN algorithm as the approach to the keyword extraction. It is viewed as a binary classification which classifies a word into keyword or non-keyword. This section is intended to describe the keyword extraction system with respect to its function and its architecture.

The process of collecting the sample words and encoding them into string vectors for implementing the keyword extraction system is illustrated in Figure 6. The keyword extraction is viewed as a binary classification which classifies a word into keyword or non-keyword. The topic-based word classification belongs to the domain independent classification where a same word is always classified identically with regardless of the domain, whereas the keyword extraction is the domain dependent classification where a same word may be classified differently depending on the domain. The words which are labeled with keyword or non-keyword are collected domain by domain. It is required to tag the input text with its own domain for executing the keyword extraction.

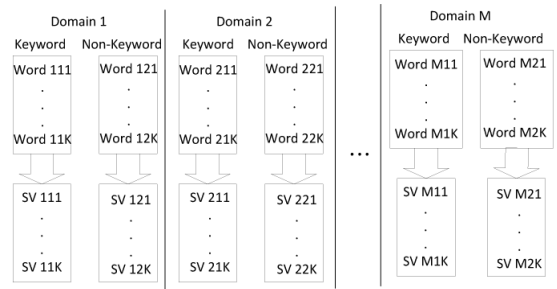


Figure 6. Preparation of Training Examples

The entire architecture of the proposed keyword extraction system is illustrated in Figure 7. A text is given as the input, and words are extracted from it in the indexing module. In the encoding module, sample words in the keyword group and the non-keyword group and ones which are indexed from the text are mapped into string vectors. The words which are indexed from the text are classified into one of the two categories in the similarity computation module and the voting module. The system consists of the four modules: indexing module, encoding module, similarity computation module, and voting module.

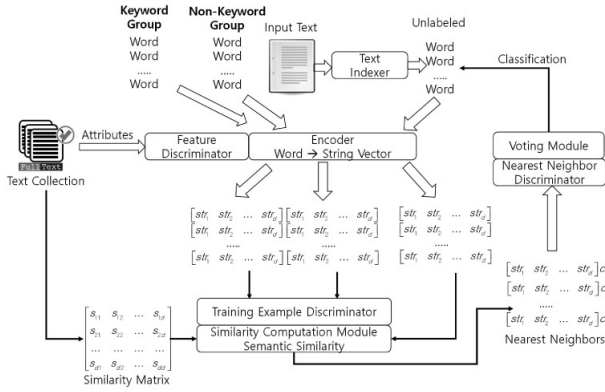


Figure 7. System Architecture

The execution process of the proposed system is illustrated as a block diagram in Figure 8. Sample words which are labeled with keyword or non-keyword are collected within a domain, and they are encoded into string vectors. An input text is indexed into a list of words, and they are also encoded into string vectors. The nearest neighbors are selected by the similarity computation, and the label is decided by voting ones of their nearest neighbors for each word. The words which are classified into keyword are extracted as the final output.

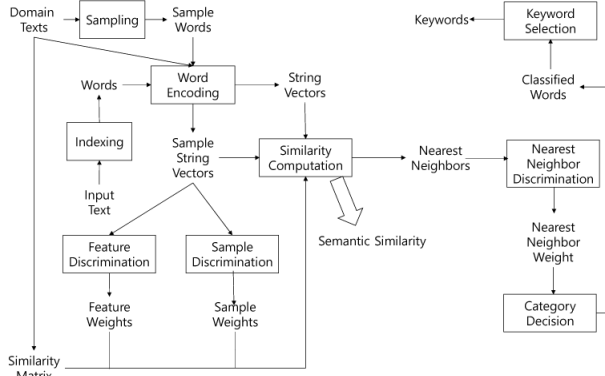


Figure 8. System Flow

Let us make some remarks on the proposed system

whose architecture is presented in Figure 7. The keyword extraction is defined as a binary classification where each word is classified into keyword or non-keyword. Each word is encoded into a string vector instead of a numerical vector and is classified directly. Among words which are included in the input text, ones which are classified into keyword are selected. We need to add the text classification module for predicting the domain of the input text automatically.

D. Connection with Text Classification

This section is concerned with the connection of the keyword extraction with the text classification. In the previous section, we mentioned the keyword extraction system as an instance of domain dependent classification. The domain is automatically decided for nominating a classifier by connecting the keyword with the text classification. The KNN algorithm which was described in Section II-B is used for implementing the text classification. This section is intended to describe the connection of the keyword extraction system with the text classification for automating the decision of the domain.

The system architecture of the text categorization system which consists of the encoding module, the similarity computation module, and the voting module is illustrated in Figure 9. The encoding module is for encoding texts into string vectors by the process which is described in Section II-A. The similarity computation module is for computing the similarities between string vectors which represent a novice text and a sample text and selecting ones with their highest similarities as the nearest neighbors. The voting module is for deciding the label of the novice item by voting the labels of the nearest neighbors. This system used for automating deciding the domain of the input text.

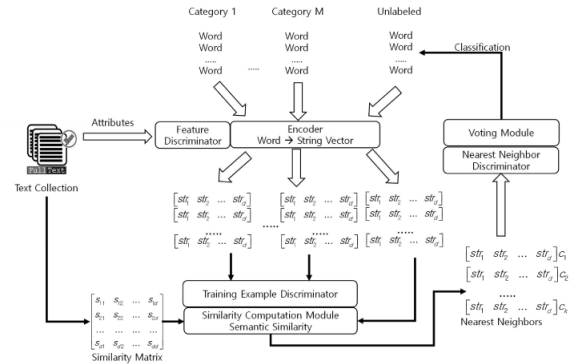


Figure 9. Text Categorization System

The connection of the keyword extraction with the text classification is illustrated in Figure 10. The texts each of which is labeled with its own domain are prepared as the sample texts, and the approach to the text classification is trained with them. A domain is assigned to a novice text by classifying it into a domain automatically. The novice

text is provided with its domain to the keyword extraction system which is described in Section II-C. The role of the text classification system is to nominate the classifier which corresponds to the domain for the keyword extraction, automatically.

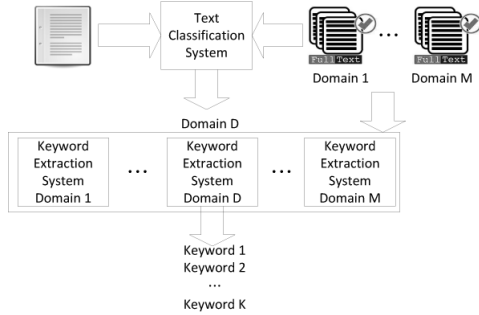


Figure 10. Keyword Extraction System connected with Text categorization

The process of executing the keyword extraction, connecting with the text classification is illustrated in Figure 11. Sample texts each of which is labeled with its own domain are prepared, and sample words each of which is labeled with keyword or non-keyword are prepared in each domain. A novice text is classified into one among the predefined domains, and the classifier which corresponds to the domain is nominated. The novice text is indexed into a list of words, and each word is classified into keyword or non-keyword. The list of words which is classified into keyword is the final output from this system; the domain which is assigned to the input text is the guide for selecting a classifier.

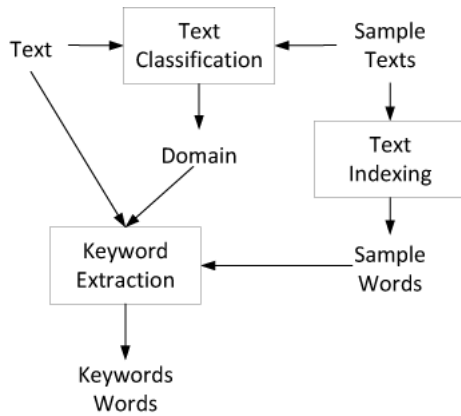


Figure 11. System Flow of Keyword Extraction connected with Text Categorization

Let us make some remarks on the connection of the keyword extraction with the text classification. The text classification is applied to the automatic decision of the input text domain. The role of text classification is to decide the domain for nominating a classifier, and the role of the keyword extraction is to extract important words from

a text. In the execution flow, the input text is classified into a domain, it is indexed into a list of words, and each word is classified into keyword or non-keyword. We consider connecting the text classification as the main task the keyword extraction as the opposite to what is proposed in this research.

REFERENCES

- [1] T. Jo, "Automatic Text Summarization using String Vector based K Nearest Neighbor", 6005-6016, Journal of Intelligent and Fuzzy Systems, 35, 2018.
- [2] T. Jo, "Modifying K Nearest Neighbor into String Vector based Version for Extracting Keywords from News Articles", 43-46, The Proceedings of International Conference on Applied Cognitive Computing, 2018.